Mobile Computing
Lecture 30
Windows CE 2

Contents

- Compatibility
- Windows CE 6.0 OAL Design
- Drivers

Compatibility

- Binary compatibility for applications is the key goal
  - Well behaved applications will work w/ little/no changes
- Compatibility maintained through CoreDLL
  - Minimize impact on Win32 APIs
  - Changes hidden in API libraries
- Apps using undocumented techniques...
  - Will likely have to be modified
  - Such as passing handles or pointers between processes
- Main changes will be in drivers and services
  - Some drivers will migrate with little work

Application Porting Test Cases

- WM 5.0 ported to Windows CE 6.0 Beta

- Running Windows CE 5.0 commercial applications on Windows CE 6.0 Beta

Compatibility Tester

- Identifies removed / deprecated / changed APIs
  - Supports both static and runtime analysis
  - Produces a detail report of any issues it finds
  - Includes documentation and suggestions

- Release before Windows CE 6.0 RTM
  - Will allow customers to prepare ahead of time

Windows CE 6 Beta BSPs

ARM	Intel Mainstone III (C-Step)	ARMv4i	Yes
	Plato VoIP Reference Platform	ARMv4i	Yes
	Device Emulator	ARMv4i	Yes
	Aruba Board	ARMv4i	No
	TI OMAP 2420	ARMv6	Yes
MIPS	NEC Rockhopper SG2 Vr5500	MIPSII & II_PF, MIPSIV & IV_FP	Yes
SH4	Hitachi/Renesas Aspen	SH4	Yes
x86	x86 (CEPC)	X86	Yes

EMB321	**Porting a Windows CE 5.0 BSP to the next release of Windows CE**	Travis Hobrla; Don Weber
EMB308	**Windows CE Secure Boot Loader**	Steve Maillet; Glen Langer

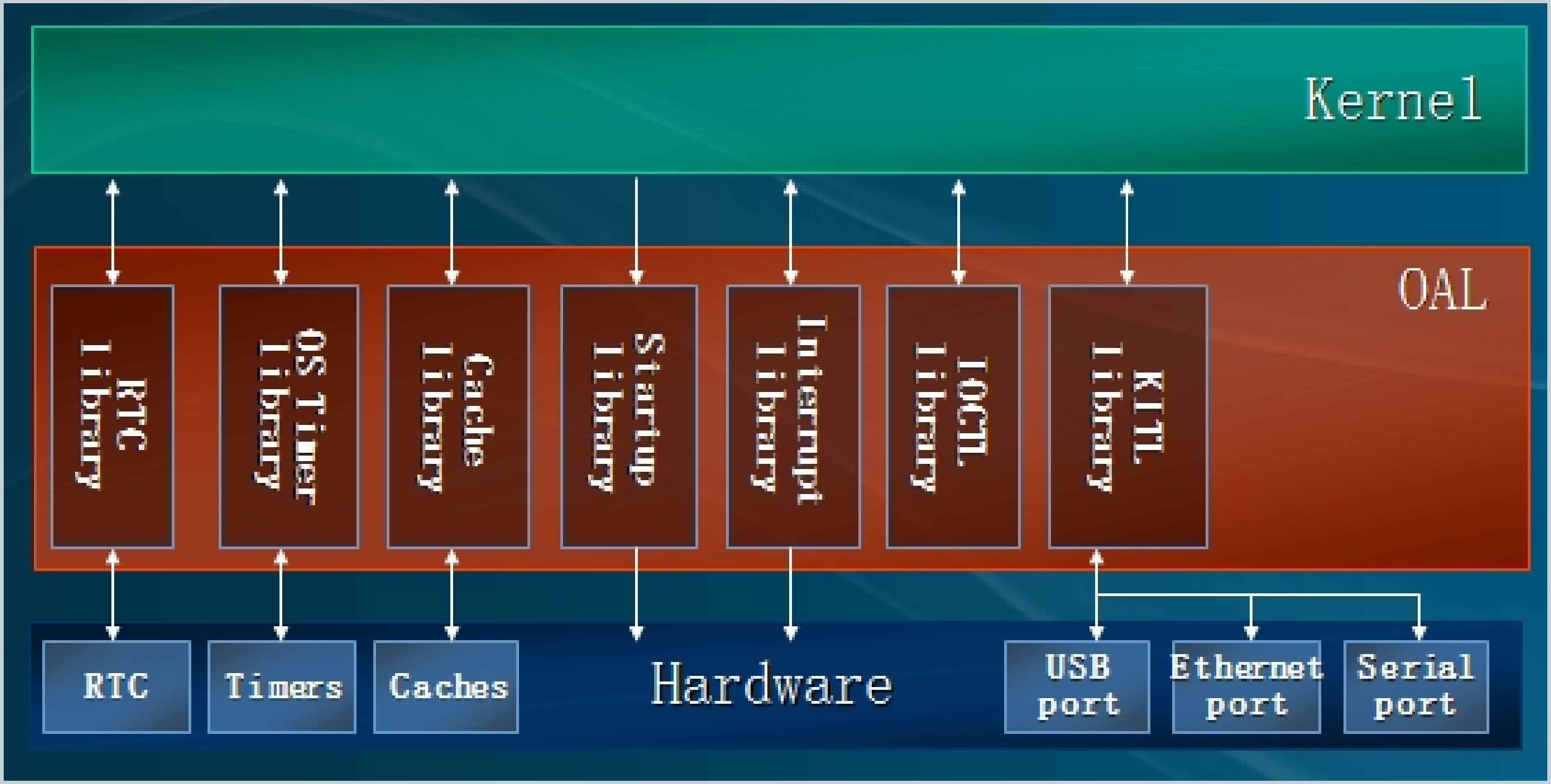
OAL Changes

- OAL split from kernel
  - Becomes “NK.EXE”
  - Kernel code becomes “Kernel.DLL”

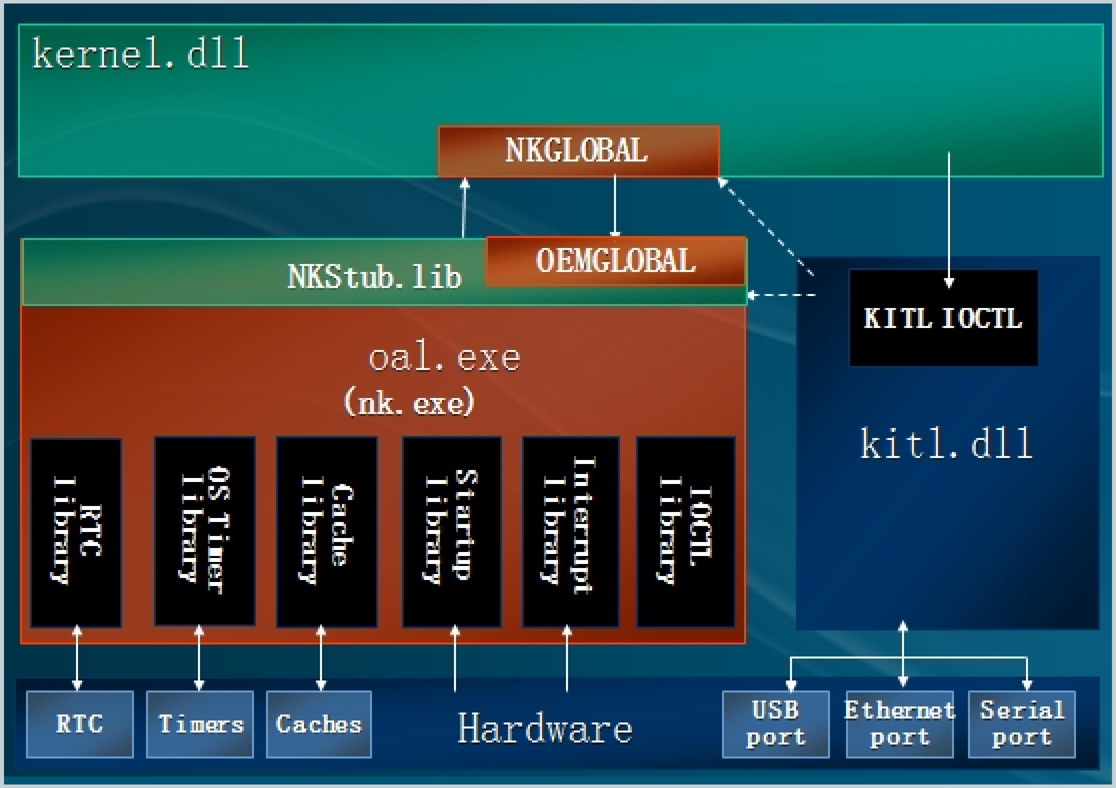
- Enables separate updates

- Overall OAL structure remains the same
  - Same OEM functions
  - OAL / kernel interface through shared structures

Windows CE 5.0 OAL Design



Windows CE 6.0 OAL Design



Drivers

- Two types of drivers will be supported
  - Kernel Mode for performance
  - User Mode for robustness
- The overall structure of the drivers remains
  - Main changes are in how the drivers access client memory
  - Drivers are still DLLs
  - Same Stream interface

Kernel Mode Drivers

- Operate in kernel’s address space
  - Calls to operating system functions very fast
  - ISRs and ISTs operate in the same process space
  - Thunking layer available for user interface services
- Drivers needing the best performance should be kernel mode
  - Such as those with lots of quick API calls

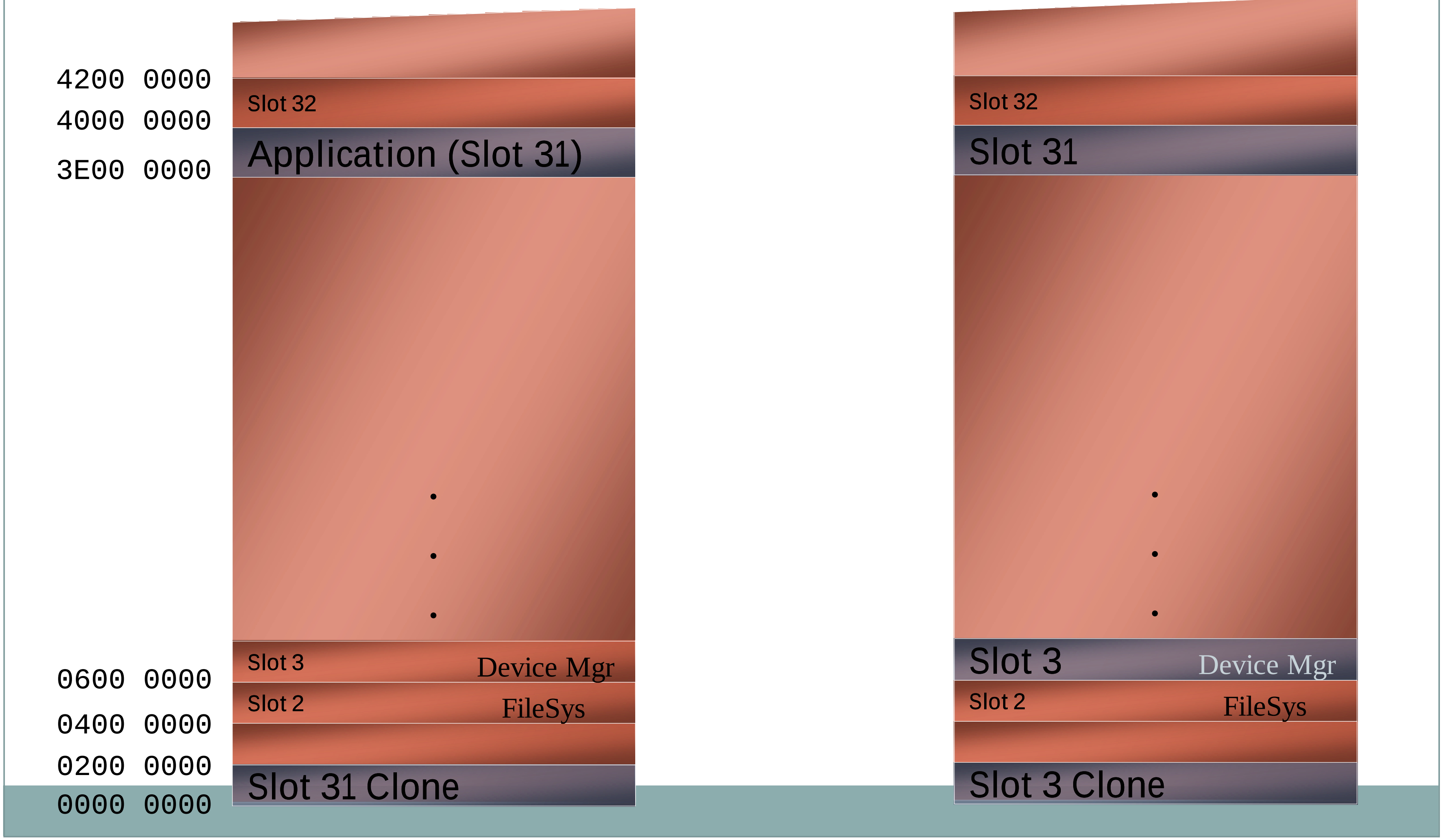
User Mode Drivers

- Loaded by udevices.exe
  - No access to kernel structures or memory
  - Same API support as applications
- Examples:
  - Expansion buses like USB and SDIO
- Drivers where performance is not a factor should consider moving to user mode
  - Called less often and do more work

Porting Drivers to the New Windows CE OS

- Most drivers become kernel mode drivers
  - Driver writers must focus on security and stability
- Maximum backward-compatibility is maintained
- Though, some driver modifications are required
  - Deprecated APIs
  - Asynchronous buffer access
  - User Interface Handling

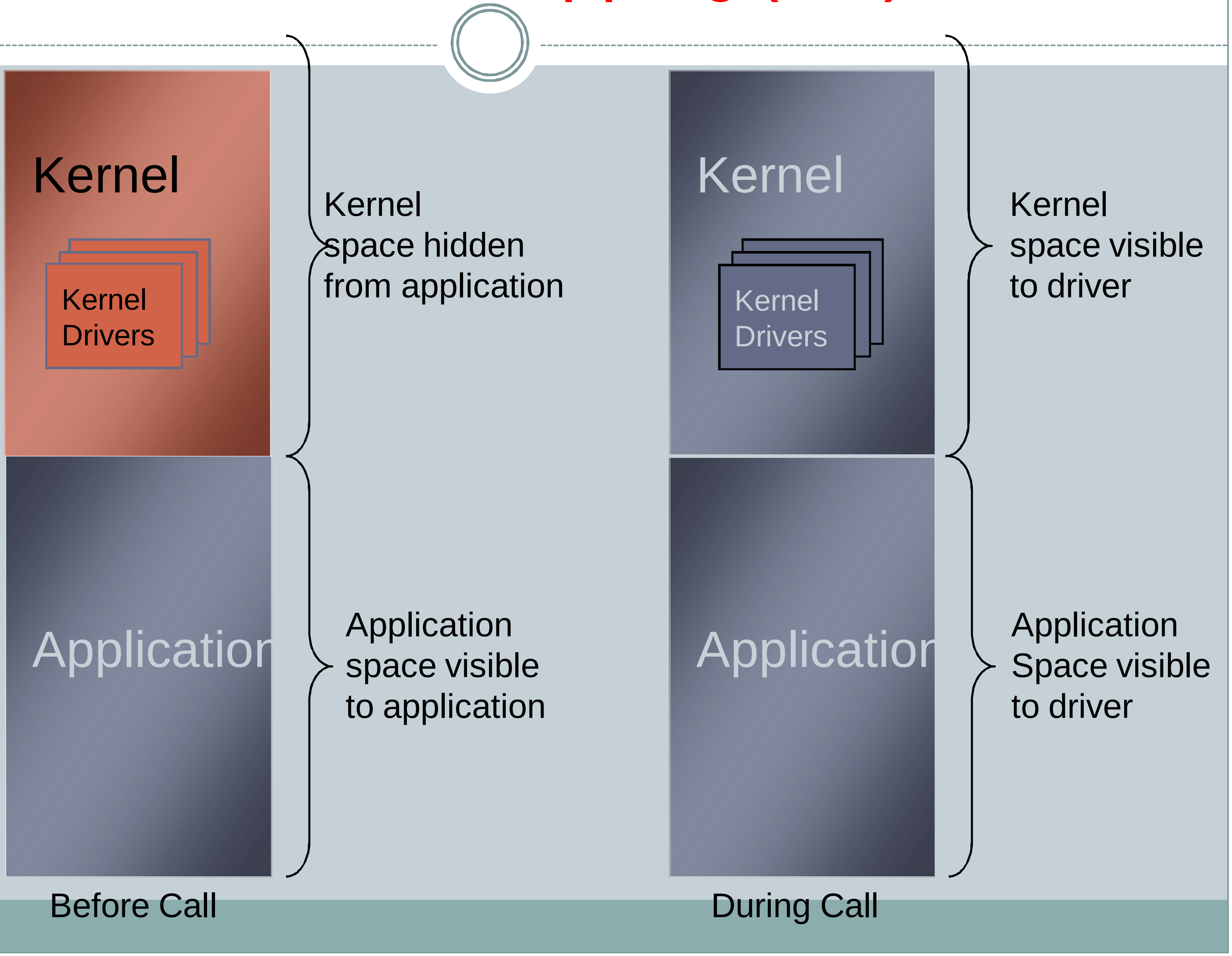
Caller Process Mapping (5.0)



At call to DeviceIoControl

When DeviceIoControl processed

Caller Process Mapping (6.0)



Kernel space hidden from application

Application space visible to application

Before Call

Kernel space visible to driver

Application Space visible to driver

During Call

Handling Calls

- App memory already mapped correctly
  - Can access it without re-mapping pointers
- Marshalling Helper Library
  - Provides APIs for handling user data
- Deprecated APIs:
  - SetProcPermissions, MapPtrToProcess, MapCallerPointer, ...

Driver Pointer Safety

- OS checks buffers referenced by caller parameters
  - Buffers are accessed checked

- Embedded pointers are valid but not access checked
  - Safe drivers should use

    CeMapCallerPointer / CeCloseCallerBuffer
  - Paranoid drivers should force duplication of buffer

Asynchronous Access

- Windows CE 6 forces new treatment of asynchronous access from driver to application

- Old:
  - SetProcPermissions to change thread access rights

- New:
  - CeAllocAsynchronousBuffer / CeFreeAsynchronousBuffer to marshal data

Summary

- Great new architecture
  - Removes the old limits
  - Performance expected as good as current
- Memory footprint similar
- OAL / Driver porting fairly straightforward